What is the cube root of 27?

Question Answering over CodeOntology

Mattia Atzeni and Maurizio Atzori

University of Cagliari (Italy)
Problem Statement

What is the **cube root** of the **max** between 20 and 27?
Problem Statement

We introduce an unsupervised approach to process questions that cannot be answered by factual QA nor advanced data querying, requiring instead ad-hoc code generation and execution.

*What is the cube root of the max between 20 and 27?*
Problem Statement

We introduce an unsupervised approach to process questions that cannot be answered by factual QA nor advanced data querying, requiring instead ad-hoc code generation and execution.

What is the cube root of the max between 20 and 27?

java.lang.Math.cbrt(java.lang.Math.max(20, 27))
Other examples

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the min between 4 and the square root of 25?</td>
<td>4</td>
</tr>
<tr>
<td>Convert &quot;mystring&quot; to upper case</td>
<td>&quot;MYSTRING&quot;</td>
</tr>
<tr>
<td>Add 2 to the cube root of 27</td>
<td>5</td>
</tr>
<tr>
<td>What is the size of &quot;abcd&quot;?</td>
<td>4</td>
</tr>
<tr>
<td>Increment 5</td>
<td>6</td>
</tr>
<tr>
<td>Raise 2 to the power of 3</td>
<td>8</td>
</tr>
<tr>
<td>Multiply 2 by the max between 1 and 3</td>
<td>6</td>
</tr>
</tbody>
</table>
Providing a semantic representation of object-oriented source code.

**1.**
- Java Project
- **OWL 2 Ontology**
- CodeOntology Parser
- RDF Triples

**2.**
- NL Question
- Semantic Parsing
- **Java**
- Execution
- Answer

International Semantic Web Conference
ISWC 2018, Monterey, Oct 10
CodeOntology (ISWC17*)

*CodeOntology* consists of two main contributions:

- an **ontology** that provides a formal representation of object-oriented programming languages;
- a **parser** capable of analyzing both Java source code and bytecode to generate RDF triples.


package org.codeontology;

public class Example {

    /** Prints a "hello world" message to the standard output */
    public static void main(String[] args) {
        System.out.println("Hello CodeOntology");
    }
}
CodeOntology

- The **ontology** is available on **Zenodo** under CC BY 4.0 license.
- The **parser** is available on **GitHub** under GPLv3 license.
- A RDF **dataset** extracted from OpenJDK 8 is available on **Zenodo** under CC BY 4.0 license.
Experiments

• CodeOntology has been applied on OpenJDK 8, gathering a structured dataset consisting of more than 2.5M RDF triples.

<table>
<thead>
<tr>
<th>Structural information</th>
<th>1981108 triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBpedia links</td>
<td>309688 triples</td>
</tr>
<tr>
<td>Source code as literals</td>
<td>134757 triples</td>
</tr>
<tr>
<td>NL Comments</td>
<td>105881 triples</td>
</tr>
</tbody>
</table>

Structural Information: 78.26%  
DBpedia Links: 12.23%  
Source Code as Literals: 5.32%  
Comments: 4.18%
SPARQL Queries

• Select all methods computing the cube root of a real number.

```
SELECT ?method
WHERE {
  ?method a woc:Method;
  woc:hasParameter/woc:hasType woc:Double ;
  dul:associatedWith dbpedia:Cube_root .
}
```

?method
java.lang.Math.cbrt(double)
java.lang.StrictMath.cbrt(double)
Question Answering over CodeOntology

We discuss two approaches:

1. a **coarse-grained** approach that only supports natural language sentences corresponding to the execution of a single method;

2. a **fine-grained** approach that supports more complex questions, possibly requiring the execution of multiple methods.
Coarse-Grained Approach

**NL specification:** "Compute the cube root"
- **Parameter:** 27.0
- **Return type:** `double.class`

```
woc:java.lang.Math.cbrt(double) 0.6
woc:java.lang.StrictMath.cbrt(double) 0.6
woc:java.lang.StrictMath.sqrt(double) 0.2
woc:java.lang.Math.sqrt(double) 0.2
...```

```
java.lang.Math.cbrt(27.0)
```

**Result:** 3.0
The ranking of Java methods relies on the following attributes:

• the name of the method
• the name of the declaring class
• the documentation comments
• semantic links to DBpedia already provided by CodeOntology
Syntactic Features

**LS:** normalized Levenshtein similarity between the natural language specification and the name of the method, computed as

\[
s_L(s_1, s_2) = 1 - \frac{d_L(s_1, s_2)}{\max \{|s_1|, |s_2|\}}
\]

where \(d_L(s_1, s_2)\) is the Levenshtein distance between \(s_1\) and \(s_2\), namely the minimum number of single character edits required to turn one string into the other.
Syntactic Features

**COM:** \( n \)-gram overlap against the Javadoc comment associated with the method, computed as

\[
ngo(S_1, S_2) = 2 \cdot \left( \frac{|S_1|}{|S_1 \cap S_2|} + \frac{|S_2|}{|S_1 \cap S_2|} \right)^{-1}
\]

where \( S_1 \) and \( S_2 \) are set of consecutive \( n \)-grams from two different sentences.

**CN:** \( n \)-gram overlap against the name of the declaring class.
Semantic Features

**W2V:** cosine similarity between the mean vectors (computed with a pre-trained Word2Vec model) associated with the natural language specification provided by the user and the Javadoc comment.

**NED:** ratio of DBpedia links shared by the method and the natural language specification.
Benchmark

We have extracted a benchmark dataset containing 122 StackOverflow questions which can be answered with the invocation of a single Java method.

Convert a string to an integer

\[
\text{java.lang=Integer.parseInt(java.lang.String)}
\]

\[
\text{java.lang=Integer.valueOf(java.lang.String)}
\]
## Experiments

<table>
<thead>
<tr>
<th>Features</th>
<th>MAP@1</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntactic Features</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LS</td>
<td>0.70</td>
<td>0.78</td>
</tr>
<tr>
<td>LS + CN</td>
<td>0.71</td>
<td>0.79</td>
</tr>
<tr>
<td>LS + COM</td>
<td>0.86</td>
<td>0.89</td>
</tr>
<tr>
<td>LS + CN + COM</td>
<td>0.87</td>
<td>0.90</td>
</tr>
<tr>
<td>Semantic Features</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NED</td>
<td>0.61</td>
<td>0.71</td>
</tr>
<tr>
<td>W2V</td>
<td>0.74</td>
<td>0.82</td>
</tr>
<tr>
<td>W2V + NED</td>
<td>0.75</td>
<td>0.82</td>
</tr>
<tr>
<td>Syntactic + Semantic Features</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LS + W2V + NED</td>
<td>0.80</td>
<td>0.86</td>
</tr>
<tr>
<td>LS + CN + COM + W2V + NED</td>
<td>0.90</td>
<td>0.92</td>
</tr>
</tbody>
</table>
Fine-Grained Approach

Given a natural language question, the fine-grained approach starts by performing Dependency Parsing.
The dependency graph is unfolded into a tree, such that:

- the set of nodes $N$ can be partitioned into two subsets, $\mathcal{L}$ and $\mathcal{M}$
- $\mathcal{L}$ is the set of literal nodes
- $\mathcal{M}$ is the set of nodes corresponding to method invocations
- Each node in $\mathcal{L}$ is a leaf
- $N = \mathcal{L} \cup \mathcal{M}$ and $\mathcal{L} \cap \mathcal{M} = \emptyset$
Mapping to a Feasible Tree

We want to apply the coarse-grained approach to label each node $i \in \mathcal{M}$ with a method ranking $\mathcal{R}_i$, that is a sequence

$$(m_1, s_1) \ldots (m_n, s_n)$$

such that

- $m_j$: Method, for each $j = 1 \ldots n$
- $s_j \in [0, 1]$, for each $j = 1 \ldots n$
- $j < k \Rightarrow s_j \geq s_k$, for each $j, k = 1 \ldots n$
Mapping to a Feasible Tree

We define the set of all types available in our knowledge base $\mathcal{K}$ as

$$\text{Types} = \{ t \in \mathcal{K} \mid t : T \land T \subseteq \text{Type} \}.$$ 

Next, we label each node with the set of types it can assume by means of a function $\text{types} : N \rightarrow 2^{\text{Types}}$, defined as

$$\text{types}(i) = \begin{cases} t & \text{if } i \in \mathcal{L} \text{ and } t \text{ is the type of } i \\ \text{returnTypes}(\mathcal{R}_i) & \text{if } i \in \mathcal{M} \end{cases}$$

$$\text{returnTypes}(\mathcal{R}) = \begin{cases} \{r\} \cup \text{returnTypes}(\mathcal{R'}) & \text{if } \mathcal{R} = \mathcal{R'}(m, s) \\ \emptyset & \text{if } \mathcal{R} = [] \end{cases}$$
Mapping to a Feasible Tree
Mapping to a Feasible Tree

We need to select a method for each ranking while maximizing the total score:

\[
\text{Maximize } \sum_{i \in \mathcal{M}} \sum_{(m_{ij}, s_{ij}) \in \mathcal{R}_i} x_{ij} \cdot s_{ij}
\]

subject to:

- \( x_{ij} \in \{0, 1\} \)
- \( \sum_{j=1}^{\mathcal{R}_i} x_{ij} = 1, \text{ for all } i \in \mathcal{M} \)
- the combination of selected methods can be compiled.
Greedy Search

We want to improve the algorithm so that it is robust to two kinds of situations:

• the tree resulting from the unfolding of the dependency graph has more nodes than needed;

• the dependency graph produced by Stanford CoreNLP contains some errors.

Hence, we perform a **greedy search** on the tree structure.
Moves

Merge
Moves

Push
Moves

MoveLiterals

A
  B
  C
    D
      F
      G
    E
  C
    D
      B
      F
      G

Objective Function

Starting from an initial tree $T_o$, the algorithm performs a greedy search with a *Best-Improvement* strategy in order to maximize, under the same constraints introduced above, the following objective function:

$$z(T_k) = \frac{1}{|M_k|} \cdot \sum_{i \in M_k} \sum_{(m_{ij}, s_{ij}) \in R^k_i} x_{ij} \cdot s_{ij} - \lambda \cdot NTED(T_k, T_0)$$

where $\lambda \in [0, 1]$ is a constant and $NTED(T_k, T_0)$ is the normalized Tree Edit Distance between $T_k$ and $T_0$, defined as

$$NTED(T, T') = \frac{TED(T, T')}{\max \{|T|, |T'|\}}$$
Experiments
<table>
<thead>
<tr>
<th>Task Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the min between 4 and the square root of 25?</td>
<td>4</td>
</tr>
<tr>
<td>Convert &quot;mystring&quot; to upper case&quot;</td>
<td>&quot;MYSTRING&quot;</td>
</tr>
<tr>
<td>Add 2 to the cube root of 27</td>
<td>5</td>
</tr>
<tr>
<td>What is the size of &quot;abcd&quot;?</td>
<td>4</td>
</tr>
<tr>
<td>Increment 5</td>
<td>6</td>
</tr>
<tr>
<td>Raise 2 to the power of 3</td>
<td>8</td>
</tr>
<tr>
<td>Multiply 2 by the max between 1 and 3</td>
<td>6</td>
</tr>
</tbody>
</table>
Experiments

The proposed approach has been compared with the WolframAlpha computational knowledge engine, on a dataset containing 120 questions on math and string manipulation.

<table>
<thead>
<tr>
<th></th>
<th>QA over CodeOntology</th>
<th>WolframAlpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Questions</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>Processed Questions</td>
<td>116</td>
<td>108</td>
</tr>
<tr>
<td>Correct Answers</td>
<td>109</td>
<td>98</td>
</tr>
<tr>
<td>Precision (global)</td>
<td>0.91</td>
<td>0.82</td>
</tr>
<tr>
<td>Precision (processed questions)</td>
<td>0.94</td>
<td>0.91</td>
</tr>
</tbody>
</table>
Experiments

Most of the execution time is spent by querying the dataset and ranking the retrieved methods. Unsurprisingly, the coarse-grained approach is faster than the fine-grained system, because it only involves the selection of one method from the dataset.

<table>
<thead>
<tr>
<th>Method Ranking</th>
<th>Instances</th>
<th>Wall-clock time (s)</th>
<th>Time per Instance (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA over OpenJDK</td>
<td>120</td>
<td>4277</td>
<td>35.6</td>
</tr>
<tr>
<td>QA over java.lang</td>
<td>120</td>
<td>718</td>
<td>6.0</td>
</tr>
<tr>
<td>Cached QA over java.lang</td>
<td>120</td>
<td>281</td>
<td>2.3</td>
</tr>
</tbody>
</table>
http://codeontology.org/
Looking for QA beta testers over Telegram! atzori@unica.it

Thank You!

Mattia Atzeni and Maurizio Atzori
3 + 2 = 5
what is the cube root of 27?

cube root(27) = 3
what is the cube root of the max between 20 and 27?

arithmetic - Cube roots of complex numbers - Mathematics Stack Exchange
https://math.stackexchange.com/.../cube-roots-of-complex-num... ▼ Traduci questa pagina
3 risposte
I need help with finding the cube roots of the complex number 27... I know that the obvious answer is three, but what is the less simple method to solving this?

Simplify a Term Under a Radical Sign - WebMath
www.webmath.com/simprad.html ▼ Traduci questa pagina
Type your term under the radical sign. The little box to the upper left of the radical sign is the power of the radical. Putting a 2 here means square root. Putting a 3 ... Mancanti: "max"

Video

Simplifying exponential expression
Khan Academy 2:37

4:20

10:53
Incrementing by 2 not 1 - C Board - Cprogramming.com
https://cboard.cprogramming.com › ... › C++ Programming ▼ Traduci questa pagina
14 ott 2004 - I'm creating a for loop that has to increment by 2 not by 1. For example something like this: for (i = 3; i < sqrt(n); i++) this adding by.

Java: Increment by 2 the two inputted integer - Stack Overflow
https://stackoverflow.com/.../java-increment-by-2-the-two-input... ▼ Traduci questa pagina
8 risposte
13 giu 2012 - 2- You are implementing the loop very well, but you just need to use...
nextInt(); System.out.println("Enter increment value: "); int increment = in.

How to increment a number by 2 in a PHP For Loop 6 risposte 7 nov 2013
javascript - Can a for loop increment/decrement by ...
7 risposte 9 ott 2012
Javascript Increment by more than 1? 4 risposte 17 mag 2012
python - How do I loop through a list by twos? 7 risposte 7 giu 2010

How can a variable increment by 2 in a while loop in C language ...
Therefore, in the absence of a specific operator to increment +2 to a variable, you will find yourself using variable=variable+k, for any constant other than 1.
Introduction

• The online availability of an increasingly large amount of open source code is dramatically changing the way programmers approach the development of large software systems.
Introduction

• This work leverages the Semantic Web technology stack to provide a semantic and queryable representation of source code.
The Ontology

• The ontology is mainly focused towards the Java programming language. It is written in OWL 2 and has been designed using the protégé resource.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>65</td>
</tr>
<tr>
<td>Object properties</td>
<td>86</td>
</tr>
<tr>
<td>Data properties</td>
<td>11</td>
</tr>
<tr>
<td>Axioms</td>
<td>1097</td>
</tr>
</tbody>
</table>
The Parser

Java Project → Dependencies Analyzer → Jar Files

Java Project → Dependencies Analyzer → Jar Files

Source Code Parser

Bytecode Parser

AST

RDF Serializer

RDF Triples
Benchmark

The image shows a bar chart and a pie chart. The bar chart displays the number of questions for relevant methods, with Method 1 having the highest number, followed by Method 2, and Methods 3 and 4 having significantly lower numbers. The pie chart further categorizes the methods with Method 1 accounting for 81.97%, Method 2 for 16.39%, and Methods 3 and 4 each for 0.82%.